Testing

EnergyPlus C++

Stuart G. Mentzer
Objexx Engineering, Inc.

C++ Compilers: Fix All Warnings

GCC: Messages are not friendly

- A lot of false positive “may be used uninitialized”

Clang: Good detection / Readable messages

The more compilers you use the better

Eliminate even false positive warnings

Static Assertions

Compile-time assertion about:

- Static object value/state
- Type attributes
- Compile-time constant expressions (constexpr)

```
static_assert( const_bool_expr, “message” );
static_assert( sizeof(Type)==42, “Bad size!” );
```

C++11 has a lot of new type testing templates:

```
static_assert( std::is_const<T>::value, “T not const” );
```

Runtime Assertions

Asserts are for coding, not user/input, errors

Assert function pre and post-conditions

No more "*the calling routine assures that*"

Assert class invariants

Assertions test *and* document expectations

Asserts are valuable when unit tests don't cover

Custom asserts that can be enabled separately

Static Testing Tools

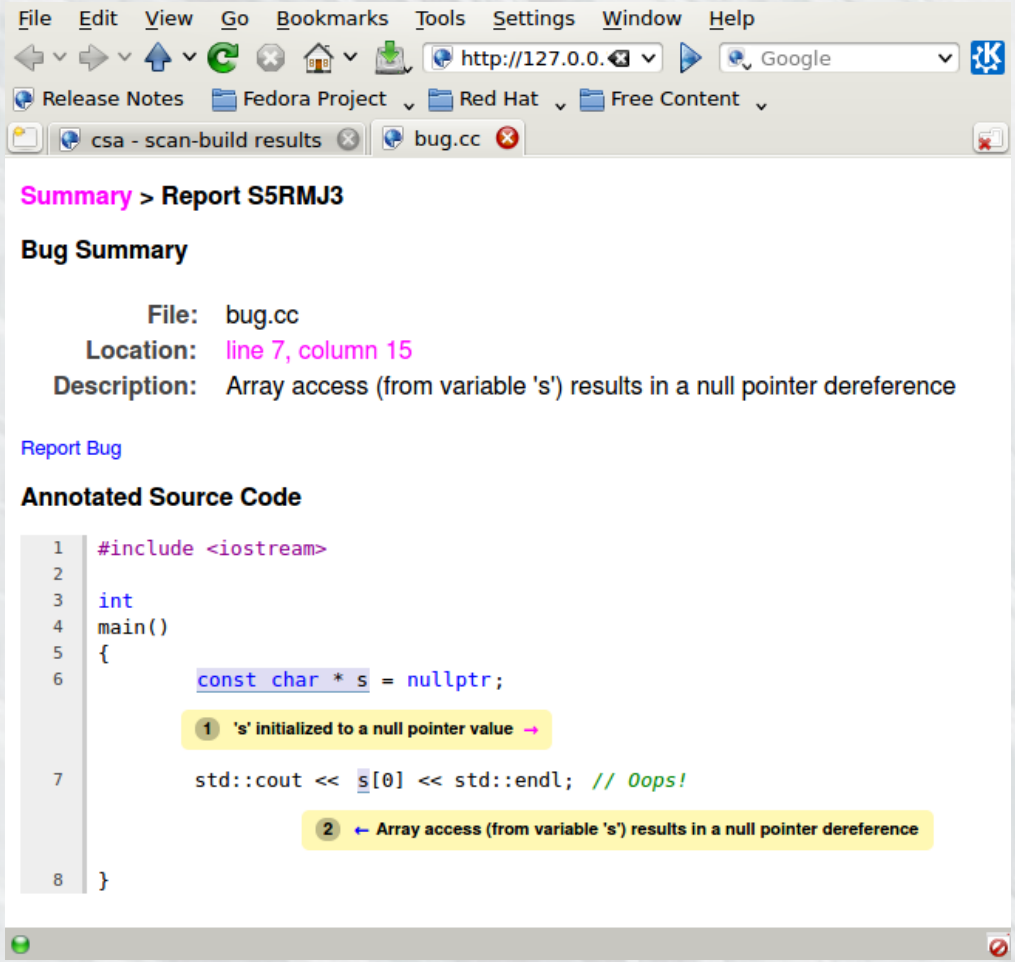
clang-analyzer
Intel Inspector
Cppcheck
CodeSonar
Coverity
C++lint
OCLint

PC-lint
PVS-Studio
QA·C++
Insure++

Find/use 2+ with high signal/noise

Clang Static Analyzer

```
scan-build clang++ -std=c++11 -c bug.cc
```



Dynamic Testing Tools

To find: memory leaks, bounds violations, use of uninitialized values,...

Address/Memory/LeakSanitizer (Clang)

valgrind

Insure++

Unit Tests for Fun and Profit

```
TEST( FmathTest, Nint )
{
  EXPECT_EQ( nint( 3.12 ), 3 );
  EXPECT_EQ( nint( 3.50 ), 4 );
}
```

Googletest

- Tests are easy to add, build, & run
- Nice colored output
- Filter to select tests to run
- Can test asserts/exceptions

```
[==========] Running 14 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 14 tests from FmathTest
[ RUN      ] FmathTest.Min
[       OK ] FmathTest.Min (0 ms)
[ RUN      ] FmathTest.Max
[       OK ] FmathTest.Max (0 ms)
[ RUN      ] FmathTest.Abs
[       OK ] FmathTest.Abs (0 ms)
[ RUN      ] FmathTest.Floor
[       OK ] FmathTest.Floor (0 ms)
[ RUN      ] FmathTest.Ceiling
[       OK ] FmathTest.Ceiling (0 ms)
[ RUN      ] FmathTest.Square
[       OK ] FmathTest.Square (0 ms)
[ RUN      ] FmathTest.Cube
[       OK ] FmathTest.Cube (0 ms)
[ RUN      ] FmathTest.Sign
[       OK ] FmathTest.Sign (0 ms)
[ RUN      ] FmathTest.Dim
[       OK ] FmathTest.Dim (0 ms)
[ RUN      ] FmathTest.Nearest
[       OK ] FmathTest.Nearest (0 ms)
[ RUN      ] FmathTest.Mod
[       OK ] FmathTest.Mod (0 ms)
[ RUN      ] FmathTest.Modulo
[       OK ] FmathTest.Modulo (0 ms)
[ RUN      ] FmathTest.Gcd
[       OK ] FmathTest.Gcd (0 ms)
[ RUN      ] FmathTest.Tolerance
[       OK ] FmathTest.Tolerance (0 ms)
[----------] 14 tests from FmathTest (171 ms total)

[----------] Global test environment tear-down
[==========] 14 tests from 1 test case ran. (249 ms total)
[  PASSED  ] 14 tests.
```

Debugging: *When All Else Fails*

Debuggers are often the worst way to find bugs

EnergyPlus is not debugger-friendly:

- Pointers to pointers to pointers
- Multi-level object containment

When more proactive methods aren't in place:

- gdb is ugly but is no longer C++ antagonistic
- GUI gdb use from Eclipse, emacs, KDevelop, ...
- Other debuggers: idb (Intel), VS, ...

Debugging: Core Dumps

Linux: Enable core file:

```
ulimit -c unlimited
```

If you get a core dump (assertion or seg fault):

```
gdb /path/to/EnergyPlus corefile
(gdb) backtrace
```

Debugging: GDB

```
gdb /path/to/EnergyPlus
(gdb) break SimulationManager.cc:367
(gdb) run
```

[stops at breakpoint]

```
(gdb) watch ('DataLoopNode::Node'.data_+98)->Temp
```

[stops when Node(99).Temp is modified]

```
(gdb) backtrace
(gdb) print i
```

Look at interesting variables

Improving Testability

Layered design of robust, modular components

Test, don’t assume, pre/post conditions

Simplify pre/run/post process

Support parallel runs (case-specific file names)

Testing (deterministic) mode:

- No time stamping outputs
- Fixed random generator seeds

Testing Policies

`static_assert` type-based conditions

`assert` pre/post-conditions & invariants

- This includes bounds testing of all containers

Unit testing requirements for new/modified code

Test mode support

Questions